Approximate Speedup by Independent Identical Processing

X. Hu, R. Shonkwiler*, and M.C. Spruill

School of Mathematics

Georgia Institute of Technology

Atlanta, GA 30332

Running head: Parallel iiP Methods

Mail proofs to:

Prof. R. Shonkwiler

School of Mathematics

Georgia Institute of Technology

Atlanta , GA 30332

*Partially supported by NSF grant G-37-653.

Abstract

In this paper we prove that for algorithms which proceed to the next state based on information available from the current state, identical independent parallel processing using stochastic multistart methods always yields a speedup in the expected time to hit a goal which is locally exponential in the number of processors.

List of Symbols

α	alpha
λ	lambda
θ	theta
μ	mu
ω	omega
η	eta
χ	chi
ξ	xi
$\mathbf{1}$	boldface one
w	lower case W

1 Introduction

Multistart methods in global optimization (see Schoen, 1990) combine a deterministic algorithm for local optimization with a randomized restart method. The randomized restart, or random choice of a new starting value for the algorithm, is applied when the algorithm has converged to a local minimum. Almost sure convergence of this method to the global minimum is assured under mild conditions; speed of convergence depends on the function being optimized, the algorithm, and the choice of restart distribution. The study of these problems by (Hu, Shonkwiler, and Spruill, 1993) included these aspects as well as the speedup available by independent identical processing (henceforth iiP) under uniform restarting. Under iiP, m processors run the algorithm simultaneously and independently. Hu, Shonkwiler, and Spruill show for certain algorithms and functions that locally exponential speedup in m is available under iiP with equi-probable restarting. The precise meaning of this phrase is revealed in equation (??) below; roughly it means that there is a constant $s > 1$ such that for moderate numbers of processors the ratio of expected time to turn the global minimum problem into a local minimum problem by m independent processors is s^{-m} times that taken

by a single processor. In this paper we prove for a completely general model of what we'll call Markov algorithms (which could be used for optimization or something completely different) and under arbitrary stationary recurrent restarting, that locally exponential speedup is always achieved by iiP. This extends our previous results to a wider class of algorithms, functions, and restarting schemes. The implication for global optimization is that for deterministic algorithms whose next step depends only on the current state (point in the domain of the function, the function's value there, the value of its gradient, or other similar functionals) iiP always results in locally exponential speedup.

Throughout this paper it is assumed that the description of the deterministic progress of the algorithm through the possible states can be made in terms of the finite union of disjoint, finite-vertexed, directed, rooted trees. The roots correspond to local minima and, whereas in the common terminology of rooted trees one typically draws them with the root on top and refers to ancestors in such a way that the root is the ancestor of all states on the tree, in our case we shall draw them root down and shall proceed down the tree to the root so that the root shall be known here as the descendent of every vertex on the tree. In this terminology, a leaf on our tree is any

vertex which has no ancestor and the unique route from such a vertex to the root shall be called a path. In the next section notation is introduced and it is shown that for iid random restarts a simpler model, the disjoint linearly ordered paths model, is sufficient for studying the time until a state in the goal is first encountered.

2 Notation and a Simple Canonical Representation

The finite set D will denote the set of states. We think of D in the case of minimization of a function f, as the domain of f. It is assumed that there is a function G from D into itself which defines on D, by identifying the points in D with vertices, and the descendent of a point $x \in D$ with the point $G(x)$, the structure of a disjoint union $\bigcup_{i=0}^{b} B_i$ of directed rooted trees B_i. The root of a tree is a point x with $G(x) = x$ and it follows from the assumptions that each B_i has exactly one root. We also refer to the subsets B_i as basins, thinking of them as the basins of attraction of a minimization algorithm applied to a function f and the roots of the B_i as the local minima

of f. Since each B_i has the structure of a rooted tree, there is for each $x \in B_i$ a unique path joining x with the root x_i of the tree and the length of this path is one less than the number of distinct elements in the set $\{G^k(x) : k \geq 0\}$. The largest such number will be attained in the tree B_i at one of its leaves and is called the height $\lambda(i)$ of the tree.

Although the results below apply to any mapping G which yields this tree structure, the situation we have in mind is when a function f on the domain D and an algorithm is specified for finding a local extremum of f. Although the algorithm could use more information at a point x than just x, for example, $(x, f(x))$ or $(x, f(x), \bigtriangledown f(x))$, it is assumed that the induced mapping G from D into D endows D with the structure of disjoint rooted trees as described above. There are many examples of algorithms which induce a function G satisfying these assumptions. One such is provided by downhill algorithms for minimization. Let f be a real valued function with domain D a finite set. Let h be any function mapping D into D which satisfies:

(i) $f(h(x)) \leq f(x)$ for all $x \in D$,

and

(ii) for each x there is a smallest $k(x) < \infty$ such that $h^k(x) = h^{k+1}(x)$ for

$k \geq k(x)$.

Then defining $H(x) = \lim_{k\to\infty} h^k(x)$ the binary relation R on $D \times D$ defined by $x\ R\ y$ if $H(x) = H(y)$ is an equivalence relation and therefore defines a partition of D into subsets, called basins, $B(f)$ and each basin $B \in B(f)$ has the structure of a directed rooted tree by defining $G(x) = h(x)$. The states in B are vertices of the tree and starting the algorithm at any state in B the algorithm proceeds down the tree, one node at a time, until it reaches the root, which is the local minimum of f in that basin. In the stochastic algorithms to be studied here, transitions occur along the branch of the tree deterministically until the root is reached whereupon the next transition is to a state in D selected at random according to a fixed probability distribution r; transitions again proceed deterministically down the branch of the tree in which the chosen state lies until the root of that tree is reached. Upon reaching the root of any tree the probability of a restart in state x is r(x) and the restart probabilities do not change over the evolution of the process. Denoting by X_n the state in D at the nth epoch, the resulting stochastic process $\{X_n : n \in \{1, 2, \ldots\}\}$ is a Markov chain with stationary transition probabilities. Defining the goal bin as the union of all basins containing the

global minimum and

$$T = \min\{n \geq 0 : X_n \text{ is in the goal bin}\}, \tag{1}$$

we shall be interested in the expected value, $E[T]$, of T. The random variable T is the number of restarts required until the algorithm receives a start at a state which will deterministically lead to the goal state.

The simplest example of a set D with the rooted tree structure is provided by the linearly ordered paths case in which each tree has exactly one leaf. In this case one can assume that

$$D = \{(i,k) : i = 0, 1, \ldots, b, k = 0, 1, \ldots, \lambda(i)\}$$

and then the Markov chain X_n has transition matrix

$$P((i,k),(i',k')) = \begin{cases} r(i',k') & \text{if } (i,k) = (i,0) \\ 1 & \text{if } i' = i,\ k' = k-1, \text{ with } 1 \leq k \leq \lambda(i) \\ 0 & \text{otherwise} \end{cases} .$$

Besides introducing some notation, the main purpose of this section is to convince the reader of the more or less obvious fact that given an arbitrary (D,G), imparting to D the rooted tree structure above, and an arbitrary distribution with $r(x) > 0$ for all $x \in D$, there is a set D^*, a function G^*

which gives D^* the linearly ordered paths structure, and a restart distribution r^* on D^* such that the probability distributions of the real valued random variables T and T^* coincide. To see that this is the case, begin with a (D, G, r) and let $\lambda(i)$ be the heights of the $b+1$ disjoint rooted trees B_i, where $D = \bigcup_{i=0}^{b} B_i$. Let

$$D^* = \{(i,k) : i = 0, 1, \ldots, b, k = 0, 1, \ldots, \lambda(i)\}.$$

Letting $k(x), x \in D$, be the smallest $k \in [0, \infty)$ such that $G^v(x) = G^{v+1}(x)$ for $v \geq k$, define

$$r^*(i,k) = \sum_{x \in A(i,k)} r(x),$$

where

$$A(i,k) = \{x \in B_i : k(x) = k\}. \tag{2}$$

Finally, set $G^*((i,j)) = (i, j-1)$ if $j \in \{1, \ldots, \lambda(i)\}$, $G^*((i,0)) = (i,0)$, and let X_n^* be the Markov chain on D^* using the initial distribution r^* defined as is X_n on D when it uses the initial distribution r.

Lemma. 2.1.

The random variables T and T^* have the same probability distribution.

Proof: Let $N(0) = 0, \ldots, N(j) = \min\{n > N(j-1) : X_n \in R\}$, where

$R = \{x_0, \ldots, x_b\}$ is the finite set consisting of the $b+1$ roots of the trees. Let $J = \min\{j \geq 0 : X_{N(j)} = x_0\}$. Then taking the sum to be zero if the upper limit is less than the lower limit

$$T = \sum_{j=0}^{J-1} (k(X_{N(j)+1}) + 1).$$

Let for $x \in D$, $i(x)$ denote the index of the tree containing x and consider the process

$$Y_n = (Y_{n1}, Y_{n2}) = (i(X_n), k(X_n)).$$

This is a Markov chain with transition matrix

$$Q((i,k),(i',k')) = \begin{cases} r(i',k') & \text{if } (i,k) = (i,0) \\ 1 & \text{if } i' = i,\ k' = k-1, \text{ with } 1 \leq k \leq \lambda(i) \\ 0 & \text{otherwise} \end{cases}$$

and the random variables $N(j)$, J, and hence T can be expressed in terms of the process Y_n by

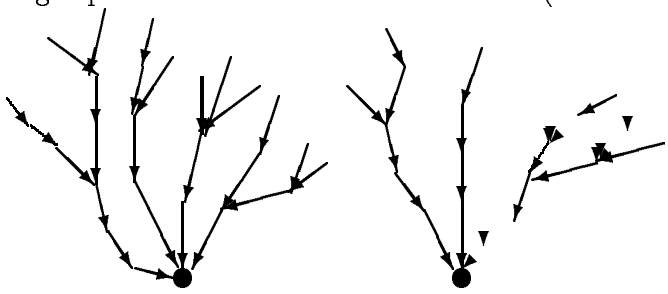
$$T = \sum_{j=0}^{J-1} (Y_{N(j)+1\ 2} + 1),$$

where $N(j) = \min\{n > N(j-1) : Y_{n2} = 0\}$ for $j \geq 1$, and $J = \min\{j : Y_{N(j)\ 1} = 0\}$. Defining the corresponding starred quantities analogously, since the state space of the process X_n^* is the same as that

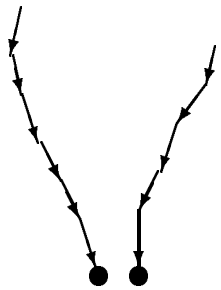
of the process Y_n, the initial distributions are the same, and the transition matrix is the same, the claim has been verified. ∎

Example 2.2.

The following represent D with its tree structure (here $b = 1$)



and a D^* with its linearly ordered tree structure associated with an equivalent chain for studying T.



3 Efficacy of Parallel Processing

The measure of efficacy of m processors employed in this paper applies to random restart methods and is defined in equation (??) as the ratio of the

expected hitting time of the goal bin by a single processor to the expected minimal hitting time of m independent processors using the same algorithm and restarting scheme. In the last section it was shown that to study the efficacy of parallel processing one need only study it for the linearly ordered paths model; this theoretical study is done in this section and a small numerical study in section 4. Let

$$D = \{(i,k) : i \in \{0,1,\ldots,b\}, k \in \{0,1,\ldots,\lambda(i)\}\}.$$

The random variable T is the number of restarts required until the algorithm receives a start at a state which will deterministically lead to the goal state. This first hitting time is defined on the Markov chain X_n whose value is the current point in D at epoch n. The random variable X_n has a probability distribution on the states which can be obtained as powers of the transition matrix P applied to the initial probability vector. The particular arrangement we choose for these states is specified in the vector of probabilities

$$\begin{aligned} p' = \quad & \big(p(0,0), p(0,1), \ldots, p(0,\lambda(0)), \\ & p(1,0), p(1,1), \ldots, p(1,\lambda(1)), \\ & p(2,0), p(2,1), \ldots, p(2,\lambda(2)), \\ & \ldots, p(b,0), p(b,1), \ldots, p(b,\lambda(b))\big), \end{aligned}$$

so the probability distribution on the states at the nth epoch is $p'P^n$ for the initial distribution p'.

Playing the key role in the calculation of $E[T]$ is the deleted $(N - n(0)) \times (N - n(0))$ transition matrix $\hat{P}$ which is the matrix which results when the rows and columns of the goal bin are deleted from P. Then, introducing the deleted $(N - n(0)) \times 1$ initial probability vector α one has

$$E[T] = \sum_{j \geq 1} P(T \geq j) = \sum_{j \geq 1} \alpha' \hat{P}^j \mathbf{1}$$

If λ is the Perron-Frobenius eigenvalue of $\hat{P}$, ω is the left eigenvector normalized so that $\omega'\mathbf{1} = 1$, χ is the right eigenvector normalized so $\omega'\chi = 1$, and $s = (\alpha'\chi)^{-1}$, then results from matrix analysis show that

$$\lim_{n \to \infty} |\frac{\alpha' \hat{P}^n \mathbf{1}}{\lambda^n} - s^{-1}| = 0. \tag{3}$$

Setting

$$T_m = \min\{k : X_k^{(j)} \text{ is in the goal bin for some } j = 1, 2, \ldots, m\}$$

for the m iid Markov chains $X_n^{(1)}, \ldots, X_n^{(m)}$, using the fact that for m iiP processors

$$E[T_m] = \sum_{j \geq 1} (\alpha' \hat{P}^j \mathbf{1})^m,$$

and carefully using the rate of the convergence of the expression in (**??**) , (Shonkwiler and Van Vleck, 1993) prove that for fixed m

$$\text{Speedup} \equiv \frac{E[T]}{E[T_m]} = s^{m-1}\frac{1-\lambda^m}{1-\lambda} + O(1-\lambda^m) \tag{4}$$

as $\lambda \to 1$. The particular form of the transition matrix P for the linearly ordered paths model enables us to prove the following.

Theorem 3.1. If all $r(i,k)$ are positive for $i \geq 0$ then the Perron-Frobenius eigenvalue λ of $\hat{P}$ is η_0^{-1}, where η_0 is the unique root greater than 1 of the polynomial

$$f(\eta) = \sum_{i=1}^{b}\sum_{k=0}^{\lambda(i)} \eta^{k+1} r(i,k) - 1. \tag{5}$$

The corresponding right eigenvector χ is proportional to v, where

$$v(i,k) = \eta_0^k,$$

and the corresponding left eigenvector ω is proportional to the vector w, where

$$w(i,k) = \sum_{u=k}^{\lambda(i)} r(i,u)\eta_0^{u-k} \quad 0 \leq k \leq \lambda(i), 1 \leq i \leq b.$$

Furthermore, defining

$$\theta_0 = \sum_{k=0}^{\lambda(0)} r(0,k),$$

if the deleted initial probability vector α is the deleted vector r then

$$s = \frac{\eta_0(\eta_0 - 1)f'(\eta_0)}{\theta_0} > \eta_0 > 1. \tag{6}$$

A proof of the theorem can be found in the Appendix. The theorem shows, for example, that with regard to parallel speedup of the expected time to hit the goal basin, the local behavior of a Markov algorithm is completely determined by the structure polynomial (**??**).

Example 3.2.

Referring to example 2.2, suppose the problem is one of minimization using a certain algorithm in which the basin B_0 containing the global minimum has the tree structure represented in Figure **??**.

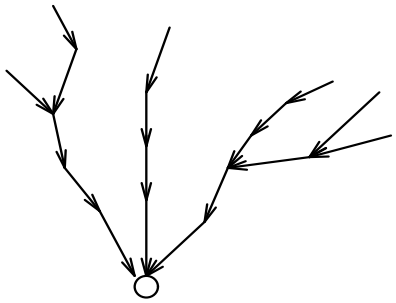


Figure 1: The corresponding tree structure of the global min.

while there is only one local, non-global, minimum which has the basin B_1 associated with Figure **??**.

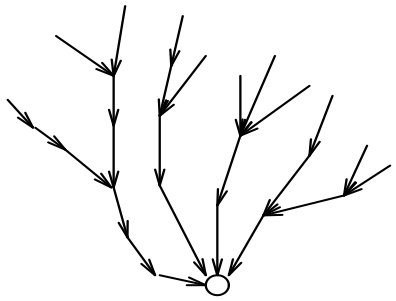


Figure 2: The corresponding tree structure of the non-global min.

Assuming equiprobable starting, $r(x) = 1/N$, where $N = 46$, one has the associated transition probabilities $r^*(i,j)$ for $i = 1$, with $r^*(1,6) = 3/46$, $r^*(1,5) = 2/46$, $r^*(1,4) = 3/46$, $r^*(1,3) = 9/46$, $r^*(1,2) = 5/46$, $r^*(1,1) = 4/46$, $r^*(1,0) = 1/46$ and the structure polynomial is

$$f(\eta) = \frac{1}{46}(\eta + 4\eta^2 + 5\eta^3 + 9\eta^4 + 3\eta^5 + 2\eta^6 + 3\eta^7) - 1.$$

It follows that for this problem in which equiprobable restarting is employed, the Perron-Frobenius eigenvalue of the deleted transition matrix is $\eta_0^{-1} = 0.8803$ and the factor s is 1.425.

Introducing the notation $n(k) = \{$number of paths of length k to the non-global minima$\}$, and letting μ be the maximal path length of any of the trees associated with non-global minima, the example above shows that

under equi-probable restarting the structure polynomial is of degree μ and is

$$f(\eta) = \frac{1}{N}\left(\sum_{j=0}^{\mu} n(k)\eta^{k+1}\right) - 1.$$

4 Some Numerical Evidence

In this section some examples involving specific algorithms are given. The first is the "successive city swap" algorithm for solving a traveling salesperson problem. The situation is simple enough so that the tree structure can actually be obtained exactly but complex enough to be too lengthy to appear here in its entirety. The structure polynomial, and therefore the factor s, is obtained precisely and predictions are made concerning the speedup due to parallel processing. To elaborate these predictions from the mathematical model, empirical evidence was gathered by conducting experiments consisting of simulations of parallel processing. Predictions from the theory are in close agreement with the empirically observed speedup.

Example 4.1.

For the TSP on 7 cities there are 720 possible tours so that $N = 720$ is the number of points in D (pairing every tour with its reverse leads to 360 distinct potential solutions but we ignore these pairings). The cities placed

in a 10×10 square have locations (0)-(2,2), (1)-(7,3), (2)-(4,5), (3)-(8,7), (4)-(1,6), (5)-(6,9), (6)-(3,8). And the space of tours is (0) 0,1,2,3,4,5,6,0; (1) 0,1,2,3,4,6,5,0; ...; (719) 0,6,5,4,3,2,1,0. The downhill algorithm employed is the "successive city swap" defined by: given a Tour (step1) start at the left, position $i = 1$, (step 2) swap the cities in positions i and $i+1$, and (step 3) if the tour length has not increased, replace Tour by this and go to step 1, else increment i and (step 4) if $i = 6$ return Tour, else go to step 2. All of the trees can be listed but only one is listed for illustrative purposes, the basin associated with the local minimum at point (6) is :

0, 120, 126, 6

128, 8, 6

264, 144, 120, 126, 6

or graphically in Figure **??**.

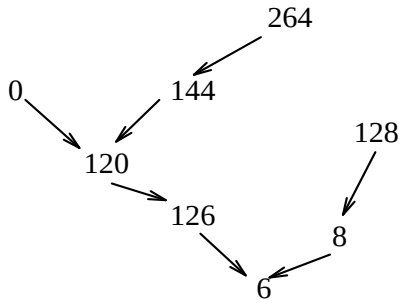


Figure 3: One of the basins of the TSP

The particular tours associated with these numbers and their lengths are as follows:

tour #	tour	length
264	3 2 1 4 5 6 0	37.67
144	2 3 1 4 5 6 0	33.98
126	2 1 4 3 5 6 0	33.06
6	1 2 4 3 5 6 0	31.01
8	1 2 4 5 3 6 0	31.71
128	2 1 4 5 3 6 0	33.06
120	2 1 3 4 5 6 0	33.48
0	1 2 3 4 5 6 0	35.32

It turns out that there are 44 basins of which 2 (123 and 478) are associated with the global minimum. The length of the longest path is 9 and the structure polynomial is

$$f(\eta) = \frac{1}{720}(42\eta+130\eta^2+174\eta^3+148\eta^4+93\eta^5+44\eta^6+18\eta^7+6\eta^8+2\eta^9+\eta^{10})-1$$

from which it follows that the Perron-Frobenius eigenvalue is $\lambda = \eta_0^{-1} = 0.9752$ and $s = 1.067$.

The algorithm was also executed 1000 times using equally likely restarting

and the complementary hitting time distribution of the resulting hitting time data is plotted in Figure ??. Fitting this curve to its theoretical form $\frac{1}{s}\lambda^{k-1}$ results in the estimates of $\lambda = 0.9728$ and $s = 1.036$.

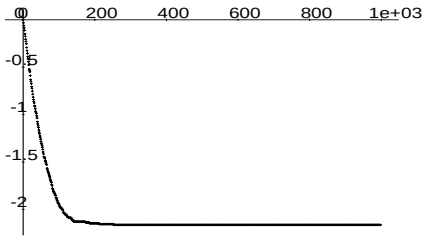


Figure 4: Complementary hitting time of the TSP problem.

Finally, parallel processing was simulated and empirical figures for the speedup were accumulated. The resulting plot is in Figure ?? and, using $\lambda = 0.9752$, indicates an acceleration $s = 1.091$.

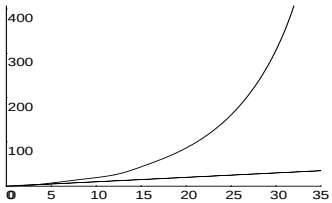


Figure 5: Speedup of the simulations of the TSP problem.

All of the estimates of speedup from Example 4.1 are in fairly close agreement considering the small sample size. The theory provides, in this simple example, predictions which fit well with empirical evidence and also illustrates that the model can be extended to the case in which the action of the algorithm depends not only the current state in D but also the iteration number. The next example, a two dimensional Shekel function, is more complicated and there are no convenient a priori predictions of the speedup available through knowledge of the polynomial.

Example 4.2.

The two dimensional shekel function is a sum of terms of the form $\frac{-1}{((x-a)^2+(y-b)^2+c)}$, so that at $x = a$ and $y = b$ this function will achieve its largest negative value of $-1/c$.

That exact values of a, b, and c are given in the table below,

a	2	9	3	6	1	8	5	2	8	9
b	1	5	8	2	3	9	8	6	5	3
c	.1	.3	.4	.3	.3	.3	.2	.4	.2	.3

A plot of the shekel function is found in Figure **??**. The algorithm employed in this example utilized the initial point to establish a direction of search and

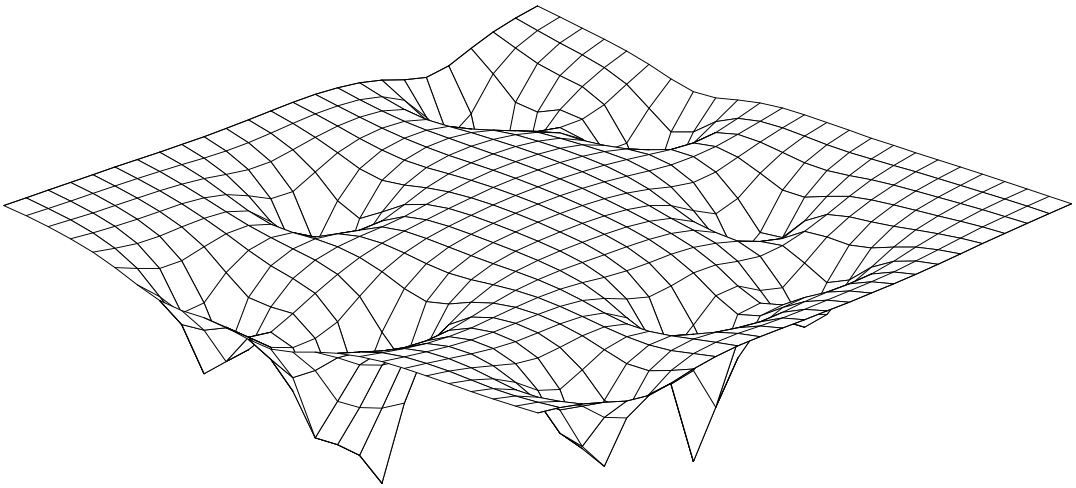


Figure 6: The two dimensional Shekel function.

conducted a line search in this direction until a settling point was reached (we're not advocating this as a good algorithm, just studying the speedup). Simulations were conducted and a plot of the speedup can be found in Figure **??**. The plot is consistent with an s factor of $s = 1.14$.

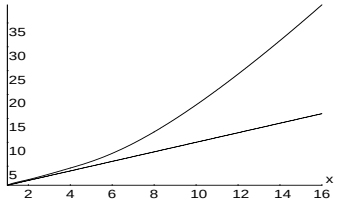


Figure 7: Speedup of the simulations of the Shekel problem.

5 Appendix

Proof of Theorem 3.1 :

First we observe that the polynomial f satisfies $f(1) = 1 - \theta(0) - 1 = -\theta(0) < 0$, $f'(\eta) > 0$, and $f''(\eta) > 0$, so that there is a unique solution $\eta > 1$ to $f(\eta) = 0$.

It is then a simple matter to verify that the given vector v is a right eigenvector and that the corresponding eigenvalue is η^{-1}, where $f(\eta) = 0$. It follows from (Varga, 1963) that η^{-1} is the P-F eigenvalue of $\hat{P}$.

Writing

$$w' = (w(1,0), w(1,1), \ldots, w(1,\lambda(1)), w(2,0), \ldots, w(b,\lambda(b)))$$

and setting $r(i,v) = w(i,v) = 0$ for $v > \lambda(i)$, one has the (i',k') coordinate

of the vector

$$w'\hat{P} = \sum_{(i,v)} w(i,v)\hat{P}((i,v),(i',k')) = \sum_{i=1}^{b}\sum_{v=0}^{\lambda(i)} w(i,v)\hat{P}((i,v),(i',k'))$$

as

$$\begin{aligned}(w'\hat{P})(i',k') &= \sum_{i=1}^{b} w(i,0)\hat{P}((i,v),(i',k')) + \sum_{i=1}^{b}\sum_{v=1}^{\lambda(i)} w(i,v)\hat{P}((i,v),(i',k')) \\ &= r(i',k')\sum_{i=1}^{b} w(i,0) + w(i',k'+1) \\ &= r(i',k')\sum_{i=1}^{b}\sum_{v=0}^{\lambda(i)} r(i,v)\eta^{v} + \sum_{v=k'+1}^{\lambda(i')} r(i',v)\eta^{v-k'-1}.\end{aligned}$$

Using $f(\eta) = 0$ on the last expression

$$\begin{aligned}(w'\hat{P})(i',k') &= r(i',k')\frac{1}{\eta} + \frac{1}{\eta}\sum_{v=k'+1}^{\lambda(i')} r(i',v)\eta^{v-k'} \\ &= \frac{1}{\eta}\sum_{v=k'}^{\lambda(i')} r(i',v)\eta^{v-k'} \\ &= \frac{1}{\eta}w(i',k').\end{aligned}$$

It has been demonstrated that the claimed eigenvalue is the Perron-Frobenius eigenvalue and the corresponding right and left eigenvectors have been found; our next step is to prove the formula for s. It is first shown that for arbitrary η

$$w'(\eta)v(\eta) = f'(\eta).$$

We have

$$\begin{aligned} w'(\eta)v(\eta) &= \sum_{i=1}^{b}\sum_{s=0}^{\lambda(i)} w(i,s)v(i,s) \\ &= \sum_{i=1}^{b}\sum_{s=0}^{\lambda(i)}(\sum_{x=s}^{\lambda(i)} r(i,x)\eta^{x-s})\eta^{s} \end{aligned}$$

and, interchanging the order of summation in the right-most sums,

$$\begin{aligned} w'(\eta)v(\eta) &= \sum_{i=1}^{b}\sum_{x=0}^{\lambda(i)}\sum_{s=0}^{x} r(i,x)\eta^{x-s+s} \\ &= \sum_{i=1}^{b}\sum_{x=0}^{\lambda(i)}(x+1)r(i,x)\eta^{x-s+s} \\ &= f'(\eta). \end{aligned}$$

The left eigenvector ω is normalized so that $\omega'\mathbf{1} = 1$. Since $\omega = aw$ one has

$$\begin{aligned} 1 = aw'\mathbf{1} &= a\sum_{i=1}^{b}\sum_{v=0}^{\lambda(i)} w(i,v) \\ &= a(\sum_{i=1}^{b}\sum_{v=0}^{\lambda(i)}(\sum_{u=v}^{\lambda(i)} r(i,u)\eta^{u-v}) \\ &= a(\sum_{i=1}^{b}\sum_{u=0}^{\lambda(i)} r(i,u)\sum_{v=0}^{u}\eta^{u-v}) \\ &= a(\sum_{i=1}^{b}\sum_{u=0}^{\lambda(i)} r(i,u)\eta^{u}\left(\frac{1-(\eta^{-1})^{u+1}}{1-\eta^{-1}}\right) \\ &= \frac{a}{\eta-1}[f(\eta)+1-\sum_{i=1}^{b}\sum_{u=0}^{\lambda(i)} r(i,u)] \\ &= \frac{a}{\eta-1}[f(\eta)+\theta_0]. \end{aligned}$$

Therefore,

$$a = \frac{\eta - 1}{f(\eta) + \theta_0}.$$

From the normalization of the right eigenvector we have from above

$$1 = \omega'\chi = aw'cv = acf'(\eta_0)$$

so that $c = 1/af'(\eta_0)$. Since

$$\begin{aligned} s^{-1} = \alpha'\chi = r'cv &= \frac{1}{af'(\eta_0)} r'v \\ &= \frac{f(\eta_0) + \theta_0}{(\eta_0 - 1)f'(\eta_0)} r'v \\ &= \left(\frac{f(\eta_0) + \theta_0}{(\eta_0 - 1)f'(\eta_0)} \right) \sum_{i=1}^{b} \sum_{u=0}^{\lambda(i)} r(i,u)\eta_0^u \\ &= \left(\frac{f(\eta_0) + \theta_0}{(\eta_0 - 1)f'(\eta_0)} \right) \frac{f(\eta_0) + 1}{\eta_0}. \end{aligned}$$

One has, using $f(\eta_0) = 0$ and dropping the subscript,

$$s = \frac{\eta(\eta - 1)f'(\eta)}{\theta_0} = \frac{\eta(\eta - 1)f'(\eta)}{-f(1)} = \frac{\eta f'(\eta)}{(f(\eta) - f(1))/(\eta - 1)} = \eta \, \frac{f'(\eta)}{f'(\xi)},$$

where the last step is a consequence of the mean value theorem and, finally, since $f'' > 0$,

$$s > \eta > 1.$$

References

[1] Schoen, F. (1991), Stochastic Techniques for Global Optimization: A Survey of Recent Advances, *J. Global Optimization* **1**, pp207-228.

[2] Hu, X., Shonkwiler, R., and Spruill, M.C. (1993), Random Restarts in Global Optimization, submitted to *SIAM J. Optimization.*

[3] Shonkwiler, R., and Van Vleck, E., Parallel Speed-up of Monte Carlo Methods for Global Optimization, to appear in *J. Complexity.*

[4] Varga, R. S. (1963), "Matrix Iterative Analysis", Prentice-Hall, Englewood Cliffs, NJ.